Using Transport Clients with Search Guard

Jochen Kressin · 2016-07-26

Search Guard supports Java transport clients to interact with a cluster. Under the hood, a transport client uses the same wire protocol as the nodes when they talk to each other. Since transport clients do not need to marshal/unmarshal JSON requests as REST clients do, they are the most efficient way of communicating with an Elasticsearch cluster.

In order for a Transport Client to talk to an Search Guard secured Elasticsearch cluster, the following requirements must be met:

- The Transport Client needs to identify itself against the cluster by sending a trusted TLS certificate
  - For that, you need to specify the location of your keystore and truststore containing the respective certificates
- A role with appropriate permissions has to be configured in Search Guard, either based on the hostname of the client, or the DN of the certificate

Add required dependencies

First, you need to add the Search Guard SSL plugin to the classpath of your application. You can download the jar file from Maven Central. Choose the version that is deployed on your cluster.

If you are using Maven, add the following entry to your pom.xml directly, and replace the sgssl.version with the Search Guard SSL version you are using.

```
<dependency>
  <groupId>com.floragunn</groupId>
  <artifactId>search-guard-ssl</artifactId>
  <version>${sgssl.version}</version>
  <scope>provided</scope>
</dependency>
```

Configure the Transport Client

In order to identify the transport client against Search Guard, add the keystore and truststore plus the respective passwords to your Settings:

```
Settings.Builder settingsBuilder =
  Settings.builder()
  .put(SSLConfigConstants.SEARCHGUARD_SSL_TRANSPORT_KEYSTORE_FILEPATH,"
<absolute/path/to/keystore>")
  .put(SSLConfigConstants.SEARCHGUARD_SSL_TRANSPORT_TRUSTSTORE_FILEPATH, "
<absolute/path/to/truststore>")
  .put(SSLConfigConstants.SEARCHGUARD_SSL_TRANSPORT_KEYSTORE_PASSWORD, "<keystore_pasword>")
  .put(SSLConfigConstants.SEARCHGUARD_SSL_TRANSPORT_TRUSTSTORE_PASSWORD, "<truststore_password>")
  ...
  Settings settings = settingsBuilder.build();
```

The keystore must contain the client's signed certificate and the CA certificate. The truststore must contain the certificate of the CA that signed the nodes certificates.

Adding the Search Guard SSL plugin

In order to activate TLS on the transport layer, put the Search Guard jar on your classpath (or add it as Maven dependency as shown above), and add the Search Guard SSL plugin to the Transport Client:

For 5.x:

```
TransportClient tc =
  new PreBuiltTransportClient(settings, SearchGuardPlugin.class)
  .<span class="pln">addTransportAddress(...)
```

For 2.x:

```
TransportClient tc =
  TransportClient
  .builder()
  .settings(settings)
  .addPlugin(SearchGuardSSLPlugin.class)
  .build()
```

Setting up authentication

Authenticating the Transport Client is no different from authenticating any other user, with one exception. Since we use the TLS certificate to prove the identity of the client, we do not need to send a password.

This is similar to using Client Authentication on the REST layer (see section Client Authentication in the Search Guard SSL docs).

You can use any authentication backend for the transport layer, but note that since we do not need a password, Search Guard only checks if the user exists in the authentication backend.

In our example, we use the internal Search Guard user database:

```
transport_auth_domain:
  enabled: true
  order: 2
  http_authenticator:
  authentication_backend:
    type: internal
```

Now we need to add the user to the sg*internal*users.yml and update the configuration. When using the Transport Client, the username is the *Distinguished Name* entry of the certificate you are using. For example, if you used the example PKI scripts (shipped with Search Guard) for generating certificates, you can use either the *kirk* or *spock* certificate. The DN of the spock certificate would be:

```
CN=spock,OU=client,O=client,L=Test,C=DE
```

If you are not sure what the DN of your certificate is, you can use the keytool shipped with the JDK to display it:

```
$ keytool -printcert -file spock.crt.pem
Owner: CN=spock, OU=client, O=client, L=Test, C=DE
Issuer: CN=Example Com Inc. Signing CA, OU=Example Com Inc. Signing CA, O=Example Com Inc.,
DC=example, DC=com
...
```

Now simply add this username to the sg*internal*users.yml file:

```
CN=spock,OU=client,O=client,L=Test,C=DE:
  hash: $2a$12$1HqHxm3QTfzwkse7vwzhFOV4gDv787cZ8BwmCwNEyJhn0CZoo8VVu
```

As being said, the password is not really necessary. But we recommend setting a strong password nevertheless, because depending on your configuration, the internal user database might be also used in other places.

Setting up permissions

Setting up the permissions for your Transport Client user is exactly the same as for any other user. Simply use the DN in the sg*roles*mapping.yml configuration file as the username, and assign it freely to any Search Guard group. In our example, we want the Transport Client to be read only. In the example configuration files shipped with Search Guard, there is already a group called “sg_readall”. Just add the DN of the certififcate as follows, and don’t forget to enclose it in double or single quotes:

```
sg_readall:
  users:
    - user1
    - user2
    - 'CN=spock,OU=client,O=client,L=Test,C=DE'
```

Now the Transport Client application is only able to read, but not to write. If we attempt to write, we get an Security Exception as expected:

```
ElasticsearchSecurityException[no permissions for indices:data/write/index]
  at com.floragunn.searchguard.filter.SearchGuardFilter.apply(SearchGuardFilter.java:163)
  ...
```

In the next article in this series, we'll show how to add usernames and passwords on transport level, so you can have use all security features of Search Guard also with a Transport Client.

This article was published on the Search Guard blog: https://search-guard.com/blog/searchguard-elasicsearch-transport-clients/